GIER SYSTEM LIBRARY

GSL no: 518
Type: Report
Author: F.B. Hermansen
I. Møller
Edition: July 1968

SORTING OF SAMBA FILES

ABSTRACT: A user-oriented description of the two associated standard procedures within the Algol 4/SAMBA system initsort which checks and stores parameters for the proper sorting procedure sort.

REF. 1: GSL. Order no 475, SAMBA - manual, june 1967.

A/S REGNECENTRALEN

Introduction

By the partition of the SAMBA-sorting-system in two procedures all parameters are checked for correct type. In GIER ALGOL 4 this is only possible for at most 6 parameters of different types in a standard procedure call.

1. initsort

```
procedure initsort (unit, ident1, ident2, fileno, crit,
number);
```

Parameter specifications:

```
value number;
integer array unit, ident1, fileno, crit;
boolean array ident2;
integer number;
```

Legality: Tape status: Irrelevant.
File status: Irrelevant.

All arrays must be one-dimentional, and unit, ident1, ident2 and fileno must all be of length n + 1 (this is tested by the procedure). n is the number of magnetic tapes which are used during the merging phase of the sorting. n must be greater than or equal to 3. (Further description of the sorting process can be found in appendix A).

In the following description we suppose that these arrays are declared from 1 through n + 1.

Consequently, the input material must be found at the tape unit with the number unit[1] on a magnetic tape which is identified with the parameters ident1[1] and ident2[1] having the same meaning as the parameters ident1 and ident2 in the standard procedure tapetest (see ref. 1, page 17).

The data material may be divided into several files in succession, in all as many as given by the parameter number. The file number of the first one is the value of the parameter fileno[1].

As mentioned before, the number of magnetic tapes which is used during the merging phase is n. They are to be mounted on the tape units with the numbers unit[2], ... , unit[n + 1]. The corresponding magnetic tapes are identified by the parameters ident1[2], ident2[2], ... , ident1[n + 1], ident2 [n + 1], and the files which may be used as working fields during the sorting have the number fileno[2], ... , fileno [n + 1].

These parameters are used by the sorting procedure, as it initializes all the magnetic tapes which have not already been handled by the procedure tapetest. If there should be any tape at some of the units which has already been initialized, it is checked that ident1 is identical to the number of the tape, (cf. ref. 1, 6.3.1, page 18).

If the parameters and tapelabel do not match, the following output will appear on the type-writer:

mount reel <ident1> on unit <unit>

after which GIER waits, and the correct tape can be mounted on the unit. Upon typing a single character on the type-writer the procedure will repeat the check until the tape is accepted and then continue.

The sorted data (records) are written on the tape unit: unit[n + 1]. Any of the n tape units on which the merging tapes are mounted may be (and this will often be the case) chosen as tape unit : unit[1], which is used for input. The tape at this unit will often function as well as input tape as merging tape.

In this case the tape number ident1[1] is identical to the corresponding parameter for the merging tape, but in case the user wants to keep the original input data, a file following the input files must be chosen as a working field.

If the user wants to secure the magnetic tape containing the input data it will appear from the parameter ident1. It is then possible to replace the input tape by a working tape as soon as the input data have been distributed to the remaining (n - 1) merging tapes during the presorting phase. At this point the following message will appear on the type-writer:

mount reel<ident1> on unit <unit>

The fifth parameter crit, (as an example declared as integer array crit[1:CR]), contains the numbers of the sorting criteria, the numerical values abs(crit[1]) abs(crit[2]), ... abs(crit[CR]) being the numbers of the words in the records which may be compared during the sorting (the record words are numbered from 1 and onwards). It is checked that none of these numbers is 0 or is greater than the smallest record length. If these demands are not met, you will get the error message:

error 17.

The comparison of two records takes place by comparing the criterion-words in crit-index-order until either the two words are different or the maximum index is reached.

The sign of the values in crit indicates whether the sorting is to take place after decreasing or increasing criterion. If the value is positive it means that, when comparing two records, the one which has the smallest value in the comparison word will be placed first, and vice versa in case of a negative criteria value. Initsort may only be called before the call of procedure sort. If it is called a second time before the call of sort, you will get the error message:

error 20.

The same error message will appear when calling sort, in case initsort has not previously been called.

2. sort

Integer procedure sort (max length, high density, real, check frequency, label array, interrupt procedure).

Parameter specifications:

```
value max length, high density, real, check frequency;
integer max length, check frequency;
boolean high density, real;
<type> array label array;
integer procedure interrupt procedure;
```

Legality: Tape status: Irrelevant.
File status: Closed for all used units
The procedure initsort must have been called.

If the second paramter high density is true, the tape units must be adjusted to write with high density; in case it is false with low density. This is checked by procedure sort, and at the same time it is checked that the magnetic tapes - which are to be used as working tapes for the merging phase of the sorting - are provided with file protection ring, whereas the input tape - in case it only function as such - is accepted both with and without file protection ring. In error cases you will get the same message as for tapetest (see ref. 1, 6.3.1 page 17).

The first parameter: max length is equal to the largest existing record length. Its size is limited by the space available in GIER's core memory (cf. Appendix B). The records may be of variable length or of constant length.

The procedure treats these two cases in the following way:

1) The records have no check word. In this case max length is used as the constant length.

2) The records are supplied with check words. In this case the record length from the check-word is used whether it is constant or not, and it is only checked that this length does not exceed max length. In error cases the message: error 17 will appear.

From the first file label of the input data, one gets the value of mode, which gives the information whether the records are provided with check-word or not, and whether check-sum has been generated. All the input files must be written in the same mode, and the sorted data are written in this mode. If the file-modes are not matching the error message: error 17 will appear.

If the input files are written with check-sum control, but without check-word per record (see ref. 1, 6.3.4 page 21) check-sum is only taken during writing the output file. The sum is delivered in "sambasum", and it is left to the user to compare this sum with a previously formed check-sum.

If the records in the input files are written with check-sum per record, check-sum control will always be performed per record during reading of the input files and during the last merging to the output file.

Moreover it is possible to have the check-sum controlled during the merging phase with a frequency determined by the value of the fourth parameter check frequency. In this case the data material will be scanned check frequency minus one times without sum-check, while sum-check is performed during the next scanning, and so on.

This control of check-sum during the merging phase may be avoided by giving the parameter either the value 0 or a very great value (however less than 1023).

When writing the records, the check-sum is simply copied, unlike outrec where it is regenerated.

In case the third parameter real is true, the content of the criteria words in the records are interpreted as real numbers. If the value of the parameter is false, they are interpreted as integers.

The fifth parameter: label array will be the user's part of the file label on the output file (compare ref. 1, 6.3.1 page 23).

However, after the final filemark only the internal file label will be written followed by a nonsense label of one word.

The sixth parameter interrupt procedure is an algol procedure written by the user. Its function must be as described in ref. 1 section 7.

Error types 1, 2, 3, and 4 may be treated as normally, but in case of error type 5 (i.e. end of file sensed) one must return from the procedure with unchanged error type.

The sorting procedure can only handle files having an accumulated length less than the capacity of one magnetic tape. Therefore, in case of error type 6 or 7 (i.e. end of tape sensed), the error procedure should give up the procedure sort by a jump, which leads out of the interrupt procedure. Thereby file status is undefined, and consequently you should close all files.

Then - if it seems reasonable - you can try to start the sorting process again by a renewed call of procedure initsort.

If the interrupt procedure has an empty body, all errors will be ignored.

On exit from sort all units will have the file status: closed.

Value

If the sorting has succeeded the value of sort is equal to 0. If a tape error has been detected in opening or closing the files during the sorting, the procedure returns with a value equal to the number of the tape unit on which the error appeared. If so, the sorting has not been completed.

Appendix A

1. Sort-record-table

The procedure initsort reserves a buffer area in which the information of the parameters is stored for later use for procedure sort.

This area looks like a record table and is treated likewise by the SAMBA-system with reference to an internal "name", which is a location in SAMBA-RS, (the slip-name is c83).

This internal table is retained in the buffer until procedure sort, after being called, has used the information. After that the area will be released. This is indicated by clearing the content of the internal "name". After this initsort may be called again, even if the sorting is interrupted at an improper moment due to errors.

2. Sorting Method

Procedure sort uses the available buffer area, and the greater this area, the faster the sorting. Consequently, the area occupied by arrays and record tables should be kept as small as possible when procedure sort is called.

A certain number of words - given by the value of the slip name b43 - is always left free to be used for the error procedure. In the existing version it is set to zero. (see Appendix C).

The sorting process may be divided into three phases:

1. presorting and distribution phase
2. merging phase
3. output phase (merging for output file).

Phase 1

The information listed below is acquired from the label of the first input file by and internal call of procedure getfile.

1. The mode in which these records are written. All input files must be written in this mode, if not, the message error 17 will be given. The same mode is used for the sorted records.

2. The size of area, the buffer area which must be reserved for reading the file. Of the remaining buffer area the procedure reserves a certain fraction (in the existing version 1/4 determined by the value of the slip name b44, see Appendix C) for double buffer area for the (n - 1) files, on which the input data is distributed during the first phase. However, the value of max length defines a lower limit of the size of these areas. This will hold, even if the input is contained in several files with "area" of various size. The remaining part of the buffer is used for presorting.

   With the chosen value of b44 you will usually get rather extended strings written in blocks of comparatively small length in the first phase of the sorting. However, when this is finished, the area used for presorting may be incorporated in the double buffer area. Therefore greater blocks are used during the merging phase, whereby the tape time is reduced.

   As presorting method "Quicksort" (Hoare, Com. ACM vol. 4 (1961), p. 321) has been chosen, partly because of the rather small size of the core store in GIER, partly because generally the records to be sorted are of constant length.

Phase 2

The merging process takes place as a polyphase sorting (Com. ACM vol. 6 (1963) p. 217), which delivers as a result the data as n - 1 sorted files, one on each of the n - 1 merging tapes. In this phase each merging tape gets 1/n of the free buffer area as double buffer area.

Phase 3

If at this time the output tape contains one of the sorted files, it is copied on to the free merging tape. Then the final merging on the output tape takes place, using as area the smaller of the following two:

1. The output area which was used during the merging phase.
2. Area from the first input file.

Appendix B

Limitations on the Record Size

In order to reduce the buffer transports a method has been chosen using n record places in the core store, where n is the number of merging tapes.

During the sorting, approximately 680 of the words in the core memory is occupied by the RS-SAMBA-version, fixed place for the sorting program, parameters for the procedure, four track places, etc.

The rest - approximately 340 words - must be shared by the RS-stack, n records - each of length max length, and the program part Compare, which occupies two locations per criterion, in excess of what has already been included into the fixed program place.

From this it appears that the maximum record length is about 100 words if only 3 tape units are used for the sorting. If greater records must be sorted this may be obtained by dividing them into part-records including the criterion words supplemented by a serial number as the last criterion.

Appendix C

Slip Reading

The slip reading takes place as described in ref. 1 appendix E - 1 with the following extensions:

When GIER gives the following message on the type-writer:

redefine SAMBA system

you may redefine the following slip names:

b41	initialized to 1
b43	initialized to 0
b44	initialized to 4

If b41 is redefined to 0 or e40 is equal to -1, the sorting procedures are not included in the translator.

b43 is the number of buffer words left to be used by the interrupt procedure. It may be redefined to any value between 0 and 1023.

b44 is used when distributing the free buffer area between output area and presorting area (see appendix A). b 44 may be redefined to any value between 2 and 100.

Comment

If the procedure sort is not used in an Algol program, it will not be included into the translated program either. This is independent of the fact whether other SAMBA procedures are used or not.

Appendix D

Example of sorting

```
begin
  integer array LABEL[1:8];
  integer n1, k, j;
  boolean in sort;
  procedure sort error(unit);
    value unit; integer unit;
    begin comment here is taken care of open/close errors, and EOT errors
       occured during the sorting;
    end sort error;
  integer procedure interrupt(unit, type, area, length);
    value unit, length;
    integer unit, type, length;
    integer array area;
    begin
      if in sort ∧ type≥5 then
        begin comment special action during sorting: ;
          if type≠5 then
            begin comment When EOT is sensed during sorting, the sorting
                must be given up;
              for j:=1 step 1 until 4 do close(j, LABEL);
              comment because this is an error-situation, the value of
                close is irrelevant;
              in sort:=false; j:=-unit;
              goto error;
            end;
          comment When type=5, the interrupt procedure must return
            without any action;
        end special sorting actions else
        begin comment Here may be taken any action as described in ref1;
        end
    end interrupt procedure;

  select(17);
  writetext(«
Place sorting-tape in reader »);
  lyn;
  select(16);
  n1:=readinteger;
  k:=readinteger;
  begin comment call initsort;
    integer array unit, ident1, fileno[1:n1], key[1:k];
    boolean array ident2[1:n1];
    for j:=1 step 1 until n1 do
      begin
        unit[j]:=readinteger;
        ident1[j]:=readinteger;
        ident2[j]:=boolean readinteger;
        fileno[j]:=readinteger;
      end;
    for j:=1 step 1 until k do key[j]:=readinteger;
    initsort(unit, ident1, ident2, fileno, key, readinteger);
  end call initsort;
  in sort:=true;
  j:=sort(readinteger, true, false, 1, LABEL, interrupt);
  in sort:=false;
  if j≠0 then
error: sort error(j);
end
```